

1 A METHOD AND APPARATUS FOR TESTING ROUTABILITY

2 Gi-Joon Nam

3 Sandor S. Kalman

4 Jason H. Anderson

5 Rajeev Jayaraman

6 Sudip K. Nag

7 Jennifer Zhuang

8
9 FIELD OF THE INVENTION

10 The present invention generally relates to analysis of
11 circuit designs, and more particularly to determining the
12 feasibility of routing a plurality of nets on a programmable
13 logic device.

14
15 BACKGROUND

16 The term "net" as used herein refers to an electrical
17 connection between components of a user's circuit design.
18 For example, one net may connect the output terminal of an
19 AND gate to the input terminal of another AND gate and to
20 the input terminal of a flip flop. An AND gate is one
21 component type, and a flip flop is another component type.
22 An "instance" is a single occurrence of a component type. A
23 net connects to a particular pin on a component instance.
24 For example, a clock net is connected to a specific clock
25 pin on a flip-flop component instance. In the context of
26 FPGAs, a pin of a component instance may correspond to the
27 input port of a configurable logic block (CLB), for example.
28 A "netlist" is a list of all the nets which connect the
29 component instances of a user's design and the pins on
30 component instances connected by each net.

31 The circuit design process generally includes the steps
32 of design entry, synthesis, optimization, device mapping,
33 and place-and-route, along with functional and timing
34 simulations to verify correctness. Determining the
35 feasibility of routing a set of nets given certain
36 constraints may be desirable in several situations in the

1 context of this process. Three examples are provided below.
2 In one example involving a given set of nets, it would be
3 useful for a user to know whether there exists a routing
4 solution for the set of nets such that no two nets are
5 electrically shorted in the solution. If no solution
6 exists, the nets may be restructured prior to place-and-
7 route.

8 In another example, a user may need to know the
9 feasibility of routing a set of nets given certain timing
10 constraints. Typically, only a subset of the list of all
11 nets are considered to be "timing-critical" nets. For
12 these timing-critical nets, it would be useful to know
13 whether there exists a routing solution that satisfies the
14 timing constraints. It would be desirable if this
15 determination could be made quickly and prior to routing the
16 set of all nets.

17 In the third example a small change is made to a
18 design. Since the user will not want to disrupt other
19 unchanged and validated sections of the design, it would be
20 useful to determine whether there exists a routing solution
21 for the nets which does not impact the unmodified sections
22 of the design.

23 An apparatus that addresses the aforementioned
24 problems, as well as other related problems, is therefore
25 desirable.

26 SUMMARY OF THE INVENTION

27 The present invention generally relates to determining
28 routing feasibility of a plurality of nets. In various
29 embodiments, an associated set of one or more routing
30 solutions are provided for each net, wherein each solution
31 specifies one or more routing resources consumed by the net.
32 A liveness Boolean function is generated having variables
33 that represent respective net/solution pairs. If there
34 exists a set of values for the variables such that at least
35

1 one of the variables for each net is logically true, then
2 the liveness function is true. An exclusivity function is
3 generated using the variables that represent the
4 net/solution pairs. If there exists at least one set of
5 values for the variables such that no resource is used by
6 more than a predetermined number of nets, then the
7 exclusivity function is true. The nets are routable using
8 the provided solutions if there exists one set of values for
9 the variables such that both the liveness and exclusivity
10 functions are true.

11 It will be appreciated that various other embodiments
12 are set forth in the Detailed Description and Claims which
13 follow.

14 15 BRIEF DESCRIPTION OF THE DRAWINGS

16 Various aspects and advantages of the invention will
17 become apparent upon review of the following detailed
18 description and upon reference to the drawings in which:

19 FIG. 1 is a flowchart of a process for determining
20 whether a set of nets is routable given certain routing
21 resource constraints, according to one embodiment of the
22 invention;

23 FIG. 2 illustrates an example net table;

24 FIG. 3 illustrates an example resource table; and

25 FIG. 4 is a flowchart of a process for generating
26 liveness and exclusivity constraints in accordance with one
27 embodiment of the invention.

28 29 DETAILED DESCRIPTION

30 Boolean Satisfiability (SAT) is a technique that is
31 used to determine whether there exists a satisfying truth
32 assignment for a Boolean function ϕ . Generally, SAT refers
33 to solving a function ϕ that is expressed in conjunctive
34 normal form (CNF), as known to those skilled in the art.

1 In the logical notation used herein, "&" represents a
2 logical AND, "V" represents a logical OR, and "~"
3 represents a logical NOT.

4 In one embodiment, the SAT technique is applied to
5 detailed netlist routing on a programmable logic device
6 (PLD). For example, a field programmable gate array (FPGA)
7 is a PLD having configurable signal-routing resources. The
8 nets in a design are realized physically by assigning
9 routing resources to each net, thereby connecting each net
10 to component instances in the desired manner.

11 FIG. 1 is a flowchart of a process for determining
12 whether a set of nets is routable given certain routing
13 resource constraints, according to one embodiment of the
14 invention. The process generally entails applying the SAT
15 technique to a Boolean function of routability constraints
16 that are generated from input net solutions and the
17 respective routing resources associated with the solutions.

18 At step 102, the nets comprising the circuit design
19 and the solution sets associated with the nets are input
20 for processing. Each net has associated therewith one or
21 more solutions, each solution comprising an associated set
22 of resources. For example, the resources associated with a
23 particular solution may include the routing resources
24 consumed by the solution.

25 In one embodiment, the input nets and solution sets
26 can be input in a text file, in which each net has the
27 following format:

28 **.net** net_name
29 **.sol** 1 resource_1 resource_2 ... resource_i
30 **.sol** 2 resource_1 resource_2 ... resource_i
31 ...
32 **.sol** m resource_1 resource_2 ... resource_k

1 .e

2 The keyword ".net" indicates the beginning of a set of
3 routing solutions for the net having the name *net_name*.

4 The keywords ".sol" indicate the beginning of the different
5 routing solutions for the net. Each routing solution

6 includes a solution name, for example, "1" and a list of

7 the routing resources used in the solution, for example,

8 *resource_1 - resource_i*. The ".e" keyword indicates the

9 end of the routing solutions for the net. Each net in the

10 input file has an entry having the format shown above.

11 Those skilled in the art will recognize that the above

12 format is but one example for inputting the routing

13 solutions and that the routing solutions could be input in

14 another form such as a database.

15 At step 104, a net table and a resource table are

16 constructed from the input nets and solution sets. Before

17 continuing with the process of FIG. 1, the net table and

18 resource table are described with reference to FIGs. 2 and

19 3, respectively.

20 FIG. 2 illustrates net table 150. Net table 150

21 includes rows associated with nets 1-*r*, respectively. Each

22 entry in a row denotes a solution for implementing the net.

23 For example, net 1 has the associated solutions 1-*i*, and

24 net 2 has the associated solutions 1-*j*. Recall from the

25 discussion of the information included in the input

26 constraint file that associated with each of the solutions

27 is a set of resources, for example, the routing resources

28 required in the solution.

29 For the purpose of formulating net table 150 into a

30 SAT problem, each of the net/solution pairs is assigned a

31 unique Boolean variable. For example, if net 1 referenced

32 as N1, and solutions 1-*i* are referenced as S1, S2, S3, and

1 Si, respectively, then the Boolean variables associated
2 with net 1 are N1S1, N1S2, N1S3, ..., N1Si. Note that the
3 notation NiSj represents a single variable.

4 FIG. 3 illustrates a resource table 180. Resource
5 table 180 is comprised of the set of routing resources used
6 by all of the solutions for all of the nets, along with the
7 net/solution pairs using the associated resources. For
8 example, if nets 1-r (FIG. 2) have solutions that require
9 resources 1-t, the resource table has respective rows for
10 resources 1-t. Each row of resource table 180 provides
11 identification of the net/solution pairs that use the
12 resource associated with the row. One way to identify a
13 net/solution pair is with the associated Boolean variable.
14 Thus, another way to view the rows of the resource table is
15 as lists of Boolean variables that correspond to the
16 net/solution pairs.

17 Returning now to FIG. 1, the net/solution pairs of
18 table 150 and the net/solution pairs in relation to the
19 resources of FIG. 3 are used to construct a Boolean
20 function to which the SAT technique can be applied. At
21 step 106, "liveness" and "exclusivity" constraints are
22 generated using the Boolean variables representing the
23 net/solution pairs and the resources used in the solutions.
24 The liveness and exclusivity constraints (L(X) and E(X))
25 are represented in CNF and are conjoined to form the final
26 routability Boolean function ($R(X) = L(X) \& E(X)$). The
27 routability function constructed from the liveness and
28 exclusivity constraints allows application of the SAT
29 technique to determine whether a layout is impossible given
30 the input net solutions.

1 Before completing the description of FIG. 1, the
2 generation of the liveness and exclusivity constraints will
3 be discussed with reference to FIG. 4.

4 FIG. 4 is a flowchart of a process for generating
5 liveness and exclusivity constraints in accordance with one
6 embodiment of the invention. The liveness constraint is
7 constructed to ensure that at least one solution for each
8 of the nets is chosen as a final legal routing solution.
9 The exclusivity constraint is constructed to ensure that no
10 routing resource is used by more nets than it can support.

11 At step 202, respective Boolean functions are
12 generated for the nets. Each Boolean function is the
13 logical OR of the Boolean variables associated with the
14 solutions that are available for the net. Each solution
15 has a single associated variable. For example, the Boolean
16 function $L(1)$ associated with net 1 of FIG. 2 is written as
17 $L(1) = N1S1 \vee N1S2 \vee N1S3 \vee \dots \vee N1Si$. At step 204, the
18 liveness constraint is generated as a Boolean function
19 $(L(X))$ by logically ANDing the Boolean functions $(L(1),$
20 $L(2), \dots, L(r))$ associated with the nets. The liveness
21 function is written as $L(X) = L(1) \& L(2) \& \dots \& L(r)$. Note
22 that $L(1), L(2), \dots, L(r)$ are referred to as Boolean clauses
23 of the Boolean function $L(X)$. Each clause corresponds to a
24 single net.

25 Steps 206, 208, and 210 are directed to generating the
26 exclusivity constraint in the form of a Boolean function
27 $E(X)$. Exclusivity is directed to ensuring that a resource
28 is not overused by more than one net. Thus, resources that
29 are used in the solutions of two or more nets are
30 identified at step 206 using resource table 180 (FIG. 3).
31 For each combination including two net/solution pairs that
32 are trying to use a particular resource, for example, net

1 i /solution j and net m /solution n , an exclusivity Boolean
2 clause is constructed to indicate that if net i /solution j
3 is to use the resource, then net m /solution n cannot use
4 the same resource. Specifically, $\sim NiSj \vee \sim NmSn$. Note that
5 exclusivity Boolean clauses are constructed only for
6 combinations of net/solution pairs involving different
7 nets, since a combination of net/solution pairs for the
8 same net would not violate exclusive use of the resource.
9 The exclusivity constraint for a resource r ($E(r)$) is the
10 conjunction of the exclusivity Boolean clauses formed for
11 combinations of net/solution pairs (e.g., $E(r) = (\sim NiSj \vee$
12 $\sim NmSn) \& (\sim NiSp \vee \sim NmSp)$).

13 At step 210, the exclusivity function for the
14 collection of resources ($E(X)$) is formed with a conjunction
15 of the exclusivity constraints of the individual resources.
16 Thus, $E(X) = E(r_1) \& E(r_2) \& E(r_3) \& \dots \& E(r_t)$. The
17 resulting exclusivity function, $E(X)$, is a Boolean function
18 that is in conjunctive normal form (CNF).

19 It will be appreciated that the present invention can
20 be applied to resources other than routing resources. For
21 example, exclusivity constraints could be used to ensure
22 that no two nets route to the same pin. This is applicable
23 in the context of an FPGA since component instances
24 typically have pins that can be swapped, which leads to a
25 situation in which a net routes to any number of pins but
26 no two nets may route to the same pin. In another
27 application, the resources may be the routing channels of
28 an FPGA instead of specific routing resources. In this
29 case, $E(X)$ is generated to ensure that only a limited
30 number of nets are permitted to route in each channel.
31 The logical relationship between the variables in the

1 exclusivity constraints will vary according to the degree
2 to which a resource can be shared between nets.

3 After the liveness and exclusivity constraints have
4 been generated, control is returned to step 108 of FIG. 1,
5 to which this description is redirected.

6 At step 108, the routability function is generated as
7 a conjunction of the liveness and exclusivity constraints
8 ($R(X) = L(X) \& E(X)$). A Boolean satisfiability process is
9 applied to the routability function at step 110 to
10 determine whether the routability function can be
11 satisfied. That is, the Boolean satisfiability process
12 determines whether there is any set of logic values (true
13 or false) for the net/solution variables which would render
14 the routability function logically true. Boolean
15 satisfiability processes are known to those skilled in the
16 art. For example, see J. P. M. Silva and K. A. Sakallah,
17 "GRASP—A New Search Algorithm for Satisfiability," Proc.
18 ACM/IEEE Int. Conference on Computer-Aided Design, Nov.
19 1996, incorporated herein by reference.

20 If the routability function can be satisfied, control
21 is directed to step 112 where the net/solution pairs that
22 solve the routability function are saved. It will be
23 appreciated that the net/solution pairs that solve the
24 routability function are those with the associated Boolean
25 variables having the value TRUE.

26 If two or more solutions have the value TRUE for the
27 same net, only one solution is selected for the final
28 routing. The selection of the solution may be made based
29 on delay and wire length factors associated with the
30 different solutions.

31 Boolean satisfiability typically finds a single
32 assignment of values to variables that satisfies the

1 Boolean function. However, for a particular set of nets,
2 there may exist multiple solution sets which can be found
3 using different, known heuristic algorithms for Boolean
4 satisfiability. If multiple solution sets are identified,
5 overall delay and wire length considerations may be used to
6 select one desirable solution set.

7 If the routability function cannot be satisfied,
8 decision step 114 tests whether the routability process
9 should be repeated with new solution sets. In one
10 embodiment, the routability process continues a selected
11 number of iterations. Alternatively, the routability
12 process can be user controlled and the process stopped at
13 the user's request. In another embodiment, the criteria
14 for stopping the routability process considers the number
15 of solutions provided for each net. For example,
16 initially, a subset of all the available solutions are
17 input to the process. If the routability is not satisfied
18 with the initial set, the number of solutions can be
19 increased for nets that are likely to be impeding the
20 finding of a solution. The process can be repeated until a
21 maximum number of solutions is reached or all possible
22 solutions have been considered.

23 If the routability process is to continue, control is
24 directed to step 116 to determine how to modify the
25 solution sets. One way to modify the solution sets is
26 through an analysis of the liveness function. For the
27 liveness function to evaluate to TRUE, each of its
28 constituent clauses must evaluate to TRUE. Since each
29 liveness clause corresponds to a single net, the number of
30 solutions for a net could be increased if the net's
31 liveness clause could not be satisfied. At step 118, the
32 solution sets are changed. In one embodiment, the results

of the analysis performed at step 116 are used in step 118 to generate additional solutions for selected ones of the nets. In another embodiment, the solution sets of all the nets are modified. Solutions for the nets can be generated automatically using known routing algorithms. For example, see C. Y. Lee, "An Algorithm for Path Connections and its Applications," IRE Transactions on Electronic Computers, Vol. EC=10, 1961, pp. 346-365, and L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," Proc. ACM/SIGDA Int. Symposium on Field Programmable Gate Arrays, 1995, pp. 111-117, both incorporated herein by reference. Once the set of solutions has been changed, processing continues at step 102 to check for routability using the new set of solutions.

The following example illustrates the process of transforming routability constraints into a SAT problem. Consider the following example input file:

```
.net net1
.sol 1 res1 res2 res3
.sol 2 res1 res4 res3
.sol 3 res1 res5 res6 res3
.e

.net net2
.sol 1 res2 res6 res5
.sol 2 res2 res3 res5
.e
```

The input file includes two different nets, net1, and net2, which have three and two different solutions, respectively. Each of the different solutions includes the specified ones of the resources res1-res6.

1 From the input file, the following net table is
2 constructed:

3 net1: net1/sol1 -> net1/sol2 -> net1/sol3
4 net2: net2/sol1 -> net2/sol2

5 The following resource table is also constructed from
6 the input file:

7 res1: net1/sol1 -> net1/sol2 -> net1/sol3
8 res2: net1/sol1 -> net2/sol1 -> net2/sol2
9 res3: net1/sol1 -> net1/sol2 -> net1/sol3 -> net2/sol2
10 res4: net1/sol2
11 res5: net1/sol3 -> net2/sol1 -> net2/sol2
12 res6: net1/sol3 -> net2/sol1

13 A first step of the transformation process is the
14 assignment of a Boolean variable to each net/solution pair,
15 as explained above. It will be appreciated that the number
16 of Boolean variables generated is $O(n)$, where n is the
17 number of net/solution pairs.

18 The liveness function for the input file is:

19 $L(X) = L(\text{net1}) \ \& \ L(\text{net2})$

20 where $L(\text{net1})$ and $L(\text{net2})$ in terms of the Boolean variables
21 are:

22 $L(\text{net1}) = N1S1 \vee N1S2 \vee N1S3$

23 $L(\text{net2}) = N2S1 \vee N2S2$

24 The exclusivity function for the input file is:

25 $E(X) = E(\text{res2}) \ \& \ E(\text{res3}) \ \& \ E(\text{res5}) \ \& \ E(\text{res6})$

26 where $E(\text{res2})$, $E(\text{res3})$, $E(\text{res5})$, and $E(\text{res6})$ in terms of
27 the Boolean variables are:

28 $E(\text{res2}) = (\sim N1S1 \vee \sim N2S1) \ \& \ (\sim N1S1 \vee \sim N2S2)$

29 $E(\text{res3}) = (\sim N1S2 \vee \sim N2S2) \ \& \ (\sim N1S1 \vee \sim N2S2) \ \& \\$
30 $(\sim N1S3 \vee \sim N2S2)$

31 $E(\text{res5}) = (\sim N1S3 \vee \sim N2S1) \ \& \ (\sim N1S3 \vee \sim N2S2)$

32 $E(\text{res6}) = \sim N1S3 \vee \sim N2S1$

1 Since resource res1 is used only in the solutions for net
2 net1, an exclusivity constraint is unnecessary for res1.
3 Similarly, since res4 is used only by the net1/sol2 pair,
4 an exclusivity constraint is unnecessary for res4.

5 The final routability function is $R(X) = L(X) \& E(X)$.
6 Application of the Boolean satisfiability process results
7 in the equation evaluating to true where N1S1 = false, N1S2
8 = true, N1S3 = false, N2S1 = true, and N2S2 = false. In
9 other words, the nets are routable using solution sol2 for
10 net net1 and using solution sol1 for net net2.

11 The present invention is believed to be applicable to
12 a variety of processes for implementing circuit designs and
13 has been found to be particularly applicable and beneficial
14 in PLDs. While the present invention is not so limited, an
15 appreciation of the present invention has been provided by
16 way of specific example nets and routing resources of PLDs.
17 Other aspects and embodiments of the present invention will
18 be apparent to those skilled in the art from consideration
19 of the specification and practice of the invention
20 disclosed herein. It is intended that the specification
21 and illustrated embodiments be considered as examples only,
22 with a true scope and spirit of the invention being
23 indicated by the following claims.

24